AMC
ASAS™ Interface for Mathcad®

Version 12.1

ANSYS, Inc.
Southpointe
275 Technology Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

© Copyright 2009 SAS IP, Inc. All Rights Reserved.
Unauthorised use, distribution or duplication is prohibited.

ANSYS, Inc. is certified to ISO 9001:2008

Revision Information

The information in this guide applies to all ANSYS, Inc. products released on or after this date, until superseded by a newer version of this guide. This guide replaces individual product installation guides from previous releases.

Copyright and Trademark Information

© 2009 SAS IP, Inc. All rights reserved. Unauthorized use, distribution or duplication is prohibited.

ANSYS, ANSYS Workbench, AUTODYN, CFX, FLUENT and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries located in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. All other brand, product, service and feature names or trademarks are the property of their respective owners.

Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. is certified to ISO 9001:2008

U.S. Government Rights

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

Third-Party Software

The products described in this document contain the following licensed software that requires reproduction of the following notices.

Formula One is a trademark of Visual Components, Inc.
The product contains Formula One from Visual Components, Inc. Copyright 1994-1995. All rights reserved.

See the legal information in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, please contact ANSYS, Inc.

Published in the U.S.A.

AMC User Manual

Update Sheet for Version 12.1

November 2009

Modifications:

The following modifications have been incorporated:

Section	Page(s)	Update/Addition	Explanation

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

TABLE OF CONTENTS

1. Overview....1-1
1.1 Introduction....1-1
1.2 Compatibility....1-1
2. Installation....2-1
2.1 Installing AMC....2-1
3. Running AMC....3-1
3.1 Using AMC Functions....3-1
4. AMC Functions....4-1
4.1 Functional Interface....4-1
4.2 Function Description....4-2
4.2.1 Description....4-2
4.2.2 Function Return Values....4-3
4.3 Available Functions....4-4
4.4 Administrative Functions....4-6
4.5 Beam Specific Functions....4-8
4.6 Element Specific Functions....4-13
4.7 Node Specific Functions....4-14
4.8 Results Database Functions....4-16
4.9 Miscellaneous Functions....4-18
4.10 Problem Solving....4-20
4.11 Restrictions....4-21
4.12 Performance....4-21
4.13 Model Updates....4-22

1. Overview

1.1 Introduction

AMC is an interface from ASAS™ to Mathcad®. It is intended to facilitate the development of Mathcad worksheets that require data from ASAS databases. The interface consists of a series of functions that can be accessed within the worksheet to recover data directly from the ASAS database. A typical example might be an application that tabulates forces in a particular member for several load cases and then undertakes a post-processing exercise.

Key features are:

- Direct access to the ASAS database information using Mathcad Function calls
- Recovery of element forces using element number and load case
- Recovery of nodal displacements/reactions using node number and load case
- Error trapping
- Simple installation
- Cross-platform interrogation of ASAS databases

1.2 Compatibility

The functions are designed to work with Mathcad Professional Version 7 and subsequent versions running under Microsoft® Windows®.

The interface program does not require an ASAS dongle or security file to operate.

2. Installation

2.1 Installing AMC

The notes below relate to Version 13.0 of Mathcad®. Earlier versions may differ slightly from the suggested installation. The *MathCAD* installation folder is normally c:\program files\mathsoft\mathcad 13, however, this may be different on your system. Due to the various different versions of Mathcad available, there are some files that need to be manually moved to complete the installation of ASAS™, these are located in your *ASAS* installation folder; if you are not confident in this procedure we suggest you consult your IT services provider.

Copy the file "AMC32.DLL" to the *MathCAD* \userefi folder from *ASAS* \amc\dll\

Copy the file "ASAS worksheet.xmct" to the *MathCAD* \Template folder from *ASAS* \amc\ Interface\mathcad13

Replace the *MathCAD*\doc\fundoc\user_EN.xml & *MathCAD*\doc\help_EN\user\userman.htm files with those provided in the amc\interface\mathcad13 folder; if you have previously edited these files or have performed a similar procedure for another software provider then you may need to merge each of these files before you can use the "Insert-Function" feature of MathCAD.

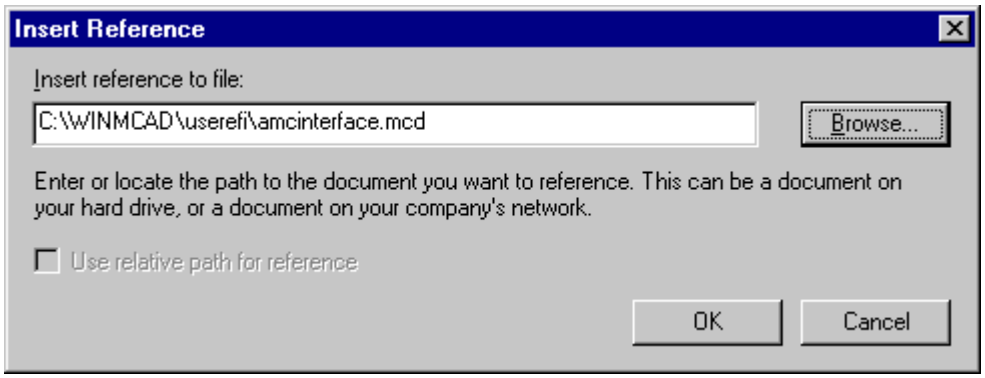
3. Running AMC

3.1 Using AMC Functions

AMC is simple to use as follows:

- Start Mathcad® as usual

- If you are creating a new worksheet use Insert Reference to make a link to the appropriate *amcinterface.xmcd* file located in your ASAS™ installation, \amc\interface\mathcadxx folder (the insertion should be undertaken at a point in the document which precedes where the AMC functions are to be utilised). This brings up a dialogue window as follows:


Insert Reference
Insert reference to file:
C:\WINMCAD\userefi\amcinterface.mcd
Browse...
Enter or locate the path to the document you want to reference. This can be a document on your hard drive, or a document on your company's network.
Use relative path for reference
OK
Cancel

The path and name of the interface file can either be typed directly, or the Browse button may be used to locate the worksheet.

- Alternatively, select ASAS worksheet from the File New Worksheet Templates. This is a simple worksheet that contains the necessary reference to the *amcinterface.mcd* file, together with some basic function calls to assist in creating the ASAS interface.

- If an existing worksheet requires the interface to be added the reference file should be inserted before the point at which the AMC functions are to be employed.

AMC functions are included in formulae just like ordinary Mathcad functions (such as sin, cos, max, etc.). If you cannot remember the order or meaning of the arguments to the function, use Mathcad's Insert Function dialogue box - the AMC functions are listed together with those supplied with Mathcad

A simple example is shown below:

amcdisplace("C:\ALPHA;PLAT;JACK",2020,401) =

This function returns the displacement vector for node 2020 in load case 401 for the specified structure JACK, in project PLAT, stored in directory C:\ALPHA.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4. AMC Functions

4.1 Functional Interface

The ASAS™ interface for Mathcad® consists of two main parts:

- a dynamic link library called amc32.dll
- a referenced Mathcad worksheet called amcinterface.xmcd

The dynamic link library (or DLL) is a set of functions written in the C language. These functions understand enough about the ASAS file system to extract information from it.

The referenced worksheet provides an interface to the functions stored on the dynamic link library (DLL) to access the ASAS file system. These functions can be used like any other functions in a Mathcad worksheet.

All the functions have names starting with the prefix *amc* and can be inserted via the insert – function menu option.

Note that the DLL functions can be called directly by omitting the reference to the interface file. This is not recommended; however, since it is not possible to pass text strings directly to and from the DLL functions, it will be necessary to undertake string conversion before and after calling the functions directly. The effort involved can be seen by double clicking on the reference icon: this will bring up the interface file. There is no advantage to be gained by using the DLL functions directly, so this should be avoided at all times.

The functions are designed to be relatively straightforward to use. All arguments are in terms of user numbers (user element number, user loadcase number, etc.) rather than ASAS internal numbers.

The AMC interface requires no special configuration, no start up macros, etc.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.2 Function Description

There are five main classes of function:

1. Administrative functions that return run time information for a particular analysis.
2. Beam specific functions that provide information that is particular to the two node beam elements.
3. Element specific functions that give information about any element type in ASAS.
4. Node specific functions that return nodal information.
5. Miscellaneous functions, returning information such as load names and numbers.

The list of functions below is complete at the time of writing, but more functions will be added as and when necessary.

4.2.1 Description

Many of the functions start with an argument called ‘model’ in the descriptions below. This argument defines the ASAS project, the structure or component being investigated, and the directory in which the files are located. It is a string consisting of three parts, separated by semi-colons:

```
directory;project;structure/component   tree
```

so, for example:

```
C:\USER;PROJ;PILE JACO
```

would specify a component with the assembled name JACO in structure PILE which is in an ASAS project called PROJ with all the files stored in a directory called \USER on drive C:. The best way of entering this data is to create a variable that is equated to the model description e.g.

```
model := "c:\asas\demo1;proj;jack"
```

The variable model can then be used in a call to any of the AMC functions that require the model name. Thus the example in section 3.1 could be rewritten as

```
model := "C:\ALPHA;PLAT;JACK"

amcdisplace(model,2020,401) =
```

4.2.2 Function Return Values

Functions may return either:

- A single value or string, such as the run title.
- A vector containing more than one value or string, such as the element forces on an element for a given loadcase.
- A matrix of results, such as the section properties for a stepped beam, where each column corresponds to a step, and each row is for a given property.

In certain circumstances a function may return either a vector or a matrix (as is the case of the function returning section properties for a beam, unstepped beams return a vector, stepped beams return a matrix). Users should be aware that Mathcad treats vectors and matrices in a different fashion, but access to these variables may be controlled by determining the number of columns of the return from a function call. This can be obtained by use of the Mathcad function cols, which returns the number of columns associated with a variable.

Thus

cols(amcbeamprop(model,1001))=

would provide how many steps are associated with beam element number 1001 in the model.

There are no explicit functions for returning the number of elements, nodes or loadcases in an analysis. This can be achieved instead by using the Mathcad function rows when retrieving appropriate lists.

Thus

rows(amcuserelementlist(model))=

would give the number of elements used in the analysis associated with model.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.3 Available Functions

FUNCTIONS	DESCRIPTION
Administrative Functions	
amcrundate	Returns analysis date and time for model
amcruntitle	Returns analysis title for model
amcprogversion	Returns the program name and version of the program utilised to create the database entry for the model
amcunits	Returns the units employed for the analysis of model
amcversion	Returns the version of the Mathcad interface
amcgetmodel	Returns the list of models that exist in a given project file
Beam Specific Functions	
amcbeamaxis	Returns beam local axes (as direction cosines) for the given element in the model
amcbeamprop	Returns beam flexural properties for the given element in the model
amcbeamoffset	Returns beam local offsets for the given element in the model
amcbeamsect	Returns beam section properties for the given element in the model
amcbeamsecttype	Returns beam section type for the given element in the model
amcbeamsectname	Returns beam section name for the given element in the model
amcbeamforce	Returns beam section forces for the given element and loadcase in the model
Element Specific Functions	
amcelementresults	Returns element forces for the given element and load in the model
amcelementtype	Returns the element type for the given element in the model
amcuserelementlist	Returns the list of user element numbers in the model
amcelementnode	Returns the node numbers associated with a given element in the model
Node Specific Functions	
amcnodenumberlist	Returns the list of user node numbers in the model
amcnodecoor	Returns the coordinates of a given node on the model
amcdisplace	Returns the global displacements for a given node and loadcase in the model
amcreaction	Returns the global reactions for a given node and loadcase in the model
Results Database Functions	
amcgetelement	Returns all the result component information for a result type at an element
amcgetelementex	Returns the requested element result, extended for additional result availability
amcgetelementflt	Returns the requested element result, filtered based on the given information
amcgetequation	Returns all the result component information for a result type at a node
amcgetglobal	Returns all the result component information for a global result type
amcgetnodal	Returns all the nodal component information for a result type at a node
Miscellaneous Functions	
amcresultant	Returns the global load resultants for a given loadcase in the model
amcgetequation	Returns the loadcase numbers that were utilised in the model
amcloadnames	Returns the loadcase names that were defined in the model
amcgroupnumber	Returns the user group numbers within a model
amcgroupelements	Returns the user element numbers within a group
amcloadtimestep	Returns the time steps of the subsets within a load case
amcmodeltimestep	Returns the time steps of loads and subsets within a model
amcnumberincrements	Returns the number of increments within a model
amcsubsetnumber	Returns the user number of the subsets within a load

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.4 Administrative Functions

Function: **amcrundate**

Purpose: Returns the date and time for a given analysis. Returned as a string.

Syntax: *amcrundate (model)*

Notes: This function returns a 17 character text string containing the date and time in the form "08.41 21-01-09".

Example: `amcrundate(Jacket)` returns the date and time information for the model defined by the variable `Jacket`.

Function: **amcruntitle**

Purpose: Returns the title used for a given analysis. Returned as a string.

Syntax: *amcruntitle (model)*

Notes: This function returns an 81 character text string containing the title in the form "Example 2.1 Extreme Wave Analysis".

Example: `amcruntitle(Jacket)` returns the title for the model defined by the variable `Jacket`.

Function: **amcprogversion**

Purpose: Returns the program name and version information associated with a particular analysis.

Syntax: *amcprogversion (model)*

Notes: This function returns a 26 character description in the form "ASAS 12.00.01.0".

Example: `amcprogversion(Jacket)` returns the program information for the model defined by the variable `Jacket`.

Function: **amcunits**

Purpose: Returns the units associated with a particular analysis. Returned as a vector of strings.

Syntax: *amcunits (model)*

Notes: The unit names are returned in the following order:

FORCE
LENGTH
ROTATION
MASS
TEMPERATURE
TIME

If units were not used in the analysis an appropriate error message is returned.

Each unit name may be up to 13 characters.

Example: `amcunits(Jacket)` returns the units used in the model defined by the variable `Jacket`.

Function: **amcversion**

Purpose: Returns a text string identifying the version of the interface worksheet file amcinterface.mcd and the associated dynamic link library amc32.dll which are currently being used. This is mainly intended so that support personnel can identify possible problems due to the use of superseded software.

Syntax: *amcversion (dummy)*

Notes: This returns a string similar to:

"AMC.MCD Version 7.00.00.0, AMC32.DLL Version 12.00.01.0"

Note that the function requires a dummy argument.

Function: **amcgetmodel**

Purpose: Returns the list of component and structure names (or models) stored in the project file. Returned as a vector of strings.

Syntax: *amcgetmodel (Directory, Project)*

Notes: *Directory* is the full path name of the directory in which the project file is stored, e.g.:

```
"C:\USER"
```

Project is the four character identifier of the ASAS project.

An entry for a component does not imply that results exist for any instance of the component.

Example: `amcgetmodel("C:\USER", "PROJ")` returns the list of components and structures stored on the project PROJ residing in the directory C:\USER.

4.5 Beam Specific Functions

Function: **amcbeamaxis**

Purpose: Returns the direction cosine components for beam element local axes. Returned as a vector.

Syntax: *amcbeamaxis (model, element)*

Notes: The direction cosine vector is returned as follows

XX
XY
XZ
YX
YY
YZ
ZX
ZY
ZZ

Where

XY corresponds to the Y component of the local X axis

The axis information returned relates to the final member orientation i.e. taking account of any rigid offsets.

Example: `amcbeamaxis(Jacket, 1001)` returns the local axes direction cosines for element 1001 in the model defined by the variable `Jacket`.

Function: **amcbeamforce**

Purpose: Returns the forces and moments on a beam element for a given loadcase. Returned as a vector.

Syntax: *amcbeamforce (model, element, loadcase)*

Notes: The forces and moments vector is returned as follows

FX_1
QY_1
QZ_1
MX_1
MY_1
MZ_1
FX_2
QY_2
QZ_2
MX_2
MY_2
MZ_2

Where

QY_1 corresponds to the Y shear force at end 1

Results for non-beam elements may be obtained with the amcelementresults function.

Example: `amcbeamforce(Jacket, 2050, 10)` returns the forces and moments for element 2050 for loadcase 10 in the model defined by the variable `Jacket`.

Function: **amcbeamoffset**

Purpose: Returns a beam element rigid offset information in terms of the element local axes. Returned as a vector.

Syntax: *amcbeamoffset (model, element)*

Notes: The offset vector is returned as follows

X_1
Y_1
Z_1
X_2
Y_2
Z_2

Where
X_1 is the offset at end 1 of the beam for the local x axis

If an offset has not been defined a value of zero will be returned.

Example: `amcbeamoffset(jacket, 1001)` returns the offsets for element 1001 in the model defined by the variable `Jacket`.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Function: **amcbeamprop**

Purpose: Returns the flexural properties for a beam element, such as length, area, inertia, etc. For unstepped beams a vector is returned, otherwise a matrix is returned with each column representing a step.

Syntax: *amcbeamprop (model, element)*

Notes: The properties for each step are returned as follows

mnemonic	Description	Applicable to [2]
L	Element or step length [1]	GRIL, BM2D, BM3D, BEAM, TUBE
A	Area	GRIL, BM2D, BM3D, BEAM, TUBE
IY	Inertia about Y axis	GRIL, BM3D, BEAM, TUBE
IZ	Inertia about Z axis	BM2D, BM3D, BEAM, TUBE
J	Torsion constant	GRIL, BM3D, BEAM, TUBE
AY	Shear area for Y axis	BM2D, BM3D, TUBE
AZ	Shear area for Z axis	GRIL, BM3D, TUBE

[1] For unstepped beams the length returned is the total physical length of the element (after the application of offsets). For stepped beams the length returned is for each step.

[2] For properties that are not applicable to a particular beam type, a value of zero will be returned.

Example: `amcbeamprop (Jacket, 1001)` returns the properties for element 1001 in the model defined by the variable `Jacket`.

Function: **amcbeamsect**

Purpose: Returns the section dimensions for a beam element. For unstepped beams a vector is returned, otherwise a matrix is returned with each column representing a step.

Syntax: *amcbeamsect (model, element)*

Notes: The properties for each step are returned as follows

mnemonic	Description	Applicable to [2]
D	diameter	TUB
T	thickness	TUB
D	depth	WF, RHS, BOX, TEE, ANGL, CHAN, PRI
TW	web thickness	WF, RHS[1], BOX[2], TEE, ANGL[1], CHAN
B	breadth	WF, RHS, BOX, TEE, ANGL, CHAN, PRI
TF	flange thickness	WF, RHS[1], BOX[3], TEE, ANGL[1], CHAN
R	fillet radius	WF, RHS, CHAN, ANGL, TEE

If section data has not been defined, an error code will be returned. For TUBE elements, however, the dimension information can still be retrieved using this function.

[1] For RHS and ANGL sections, only one thickness is defined. Both TW and TF are set to this value.

[2] For BOX sections TW corresponds to the thickness of the side plates.

[3] For BOX sections TF corresponds to the thickness of the top and bottom plates.

Example: `amcbeamsect (jacket, 1001)` returns the section properties for element 1001 in the model defined by the variable `Jacket`.

Function: **amcbeamsectname**

Purpose: Returns the section name(s) associated with a beam element. For unstepped beams a string is returned, otherwise a vector of strings is returned.

Syntax: *amcbeamsectname (model, element)*

Notes: If section data has not been defined an error code will be returned.

Each section name recovered may be up to 12 characters long.

Example: `amcbeamsectname(jacket, 1001)` returns the section name associated with element 1001 in the model defined by the variable `Jacket`.

Function: **amcbeamsecttype**

Purpose: Returns the section profile type(s) for a beam element.

Syntax: *amcbeamsecttype (model, element)*

Notes: If section data has not been defined an error code will be returned.

Profiles returned will be one of the following:

mnemonic	Description
WF	Wide Flange
TUB	Tube
BOX	Fabricated Box
RHS	Rolled Hollow Section
TEE	Tee
ANGL	Angle Section
CHAN	Channel
PRI	Prismatic

Currently ASAS™ only permits stepped beams to consist of a common section type so this function will always return the same type for each step.

Example: `amcbeamsecttype(jacket, 1001)` returns the section type of element 1001 in the model defined by the variable `Jacket`.

4.6 Element Specific Functions

Function: **amcelementnode**

Purpose: Returns the node numbers on an element. Returned as a vector.

Syntax: *amcelementnode (model, element)*

Purpose: Nodes are returned in the same order as they are defined in the ASAS data.

Example: `amcelementnode(Jacket, 2050)` returns the node numbers on element 2050 in the model defined by the variable `Jacket`.

Function: **amcelementtype**

Purpose: Returns the ASAS™ element type. Returned as a string.

Syntax: *amcelementtype (model, element)*

Notes: This function returns the four character name by which an element type is identified within the ASAS system e.g TUBE, BM3D, QUM4, etc.

Example: `amcelementtype(Jacket, 1001)` returns the ASAS element type for user element 1001 in the model defined by the variable `Jacket`.

Function: **amcelementresults**

Purpose: Returns forces or stresses for any type of element.

Syntax: *amcelementresults (model, element, loadcase)*

Notes: All the results for one load case on an element are stored as a list of numbers by ASAS. It is necessary to know the order of the results and the node numbering of the element to extract the required result. The amcbeamforce function should be used for retrieving results for the standard three-dimensional beam elements.

Example: `amcelementresults(vessel, 2050, 10)` returns the stresses associated with element number 2050 for loadcase 10 in the model defined by the variable `vessel`.

Function: **amcuserelementlist**

Purpose: Returns the list of user element numbers.

Syntax: *amcuserelementlist (model)*

Notes: The element numbers are returned in ascending order. Components are excluded from this list. The number of elements in the model may be found using the Mathcad function *rows* e.g.

```
NumberOfElements:=rows(amcuserelementlist(model))
```

Example: `amcuserelementlist(Jacket)` returns the user element numbers for the model defined by the variable `Jacket`.

4.7 Node Specific Functions

Function: **amcnodenumberlist**

Purpose: Returns the list of user node numbers. Returned as a vector.

Syntax: *amcnodenumberlist (model)*

Notes: The node numbers are returned in ascending order. Non-structural nodes (those not connected to elements) are excluded. The number of nodes in the model may be found using the Mathcad function *rows*. E.g.

```
NumberOfNodes:=rows(amcnodenumberlist(model))
```

Example: `amcnodenumber(Jacket)` returns the node numbers for the model defined by the variable `Jacket`.

Function: **amcnodecoor**

Purpose: Returns the global co-ordinates of a specified node. Returned as a vector.

Syntax: *amcnodecoor (model, nodenumber)*

Notes: Co-ordinates that were defined in the ASAS data in a system other than the global Cartesian system are converted prior to storage on the ASAS database. The original definition cannot be retrieved.

Example: `amcnodecoor(Jacket, 1010)` returns the global co-ordinates for node 1010 in the model defined by the variable `Jacket`.

Function: **amcdisplace**

Purpose: Returns the global displacements at a node for a specified load case. Returned as a vector.

Syntax: *amcdisplace (model, node, loadcase)*

Notes: The displacements are returned in the same order that they are reported in ASAS. If the node is skewed, the displacements are in the skewed directions.

Example: `amcdisplace(Jacket, 1010, 10)` returns the displacement vector at node 1010 for loadcase 10 in the model defined by the variable `Jacket`.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Function: **amcreaction**

Purpose: Returns the global reactions at a node for a specified loadcase. Returned as a vector.

Syntax: *amcreaction (model, node, loadcase)*

Notes: The reactions are returned in the same order that they are reported in ASAS. If the node is skewed, the reactions are in the skewed directions

Example: `amcreaction(Jacket, 1010, 10)` returns the reaction vector at node 1010 for loadcase 10 in the model defined by the variable `Jacket`.

4.8 Results Database Functions

The following four functions can only be used with the new Unified Results Storage Database. Therefore it is necessary to include the option RESU in the ASAS, ASAS-NL, etc data file. This will then produce a physical 45 file which the functions below will extract information from.

Function: **amcgetelement**

Purpose: Returns all the result component information for a result type at an element. Returned as a vector.

Syntax: *amcgetelement (model, element, loadcase, subset, resulttype)*

Notes: The components are returned in the same order as they are saved in ASAS.

Example: amcgetelement(`Jacket`, 2, 10, 3, "stress") returns the stresses applied at element 2 for the third subset of loadcase 10 in the model defined by the variable `Jacket`.

Function: **amcgetelementex**

Purpose: Returns the requested element result, extended for additional result availability

Syntax: *amcgetelementex (model, element, load, subset, result type, result component, surface, node, result position)*

Notes: This is an extended version of amcgetelement, all input is the same as that, except for the result component, additional surface, node and result position which means that it acts the same as the function in AXL and returns a single value. The additional parameters identify the particular component, the surface on which it is to be obtained and the local element node number (1 to max number of nodes on the element) and an optional result position, which is used for results where there are multiple values of a result component, for example, STRESS in a time history based FATJACK analysis with rainflow counting.

Example: `amcgetelementex (c:\;ABCD;WXYZ, 72, 11, 1, “RANGE HISTOGRAM”, “STRESS”,1,2,10)` returns the 10th STRESS in the RANGE HISTOGRAM values at the second node on the first surface of element 72. The results are retrieved for the 11 loadcase, 1st subset within the project ABCD, structure WXYZ, located in C:\.

Function: **amcgetelementflt**

Purpose: Returns the requested element result, filtered based on the given information.

Syntax: *amcgetelementflt (model,element, load, subset, result type, result component, surface, node, result position, filter type)*

Notes: This is an extended version of amcgetelementex however it allows for the filtering of a range of values to be obtained. Model, result type, result component and result position all function as per amcgetelementex. The filtering is controlled by the filter type and requested value, for which the following should be entered:

Filter type	Filtering algorithm
1	Maximum
2	Minimum
3	Absolute Maximum (i.e. furthest from zero)
4	Absolute Minimum (i.e. closest to zero)

Element, load, subset, surface and node should have either a particular value assigned to limit the searching to a particular set of results, or -1 entered to loop over each result available, e.g. entering a loadcase of -1 will mean that all loadcases will be searched according to the filter type and requested value.

The retuned array will have rows of results as follows

Return type	Returned Result
1	Value of Result
2	Element Number at which the value can be found
3	Load Case / Set for which the value can be found
4	Load Subset for which the value can be found
5	Surface at which the value can be found
6	Local node at which the value can be found

Example: `amcgetelementflt(c:\;ABCD;WXYZ, -1, 100, 0, “API LRFDALLOED1 UC”, “UC.AXIAL”, 1,-1,1,2)` returns the array of values for all positions on all elements for loadcase 100. The results are retrieved for the project ABCD, structure WXYZ, located in C:\

Function: **amcgetequation**

Purpose: Returns all the result component information for a result type at a node. Returned as a vector.

Syntax: *amcgetequation (model, node, loadcase, subset, resulttype)*

Notes: The components are returned in the same order as they are saved in ASAS.

Example: amcgetequation(`Jacket`, 1010, 10, 3, "displacement") returns the displacements for node 1010 for the third subset of loadcase 10 in the model defined by the variable `Jacket`.

Function: **amcgetglobal**

Purpose: Returns all the result component information for a global result type. Returned as a vector.

Syntax: *amcgetglobal (model, loadcase, subset, resulttype)*

Notes: The components are returned in the same order as they are saved in ASAS.

Example: amcgetglobal(`Jacket`, 10, 3, "REACTION SUM") returns the global reactions for the third subset of loadcase 10 in the model defined by the variable `Jacket`.

Function: **amcgetnodal**

Purpose: Returns all the nodal result component information for a result type at a node. Returned as a vector.

Syntax: *amcgetnodal (model, node, loadcase, subset, resulttype)*

Notes: The components are returned in the same order as they are saved in ASAS.

Example: amcgetnodal(`Jacket`, 3, 10, 11, "history displacement") returns the history displacement of node 3 for the 11th subset of loadcase 10 in the model defined by the variable `Jacket`.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.9 Miscellaneous Functions

Function: **amcresultant**

Purpose: Returns the global load resultants for a specified load case. Returned as a vector.

Syntax: *amcresultant (model, loadcases)*

Notes: The total load resultant is that calculated by ASAS during the data check. Hence it will be inaccurate if the resultant load calculated during the data check is inaccurate.

Example: `amcresultant(Jacket, 1010)` returns the load resultant for loadcase 1010 in the model defined by the variable `Jacket`.

Function: **amcloadnames**

Purpose: Returns the titles of the loadcases analysed. Returned as a vector of strings.

Syntax: *amcloadnames (model)*

Notes: If only one loadcase has been analysed then this function returns a variable.

Example: `amcloadnames(Jacket)` returns the loadcase titles in the model defined by the variable `Jacket`.

Function: **amcloadnumbers**

Purpose: Returns the user loadcase numbers employed in an analysis. Returned as a vector.

Syntax: *amcloadnumbers (model)*

Notes: If only one loadcase has been analysed then this function returns a variable.

Example: `amcloadnumbers(Jacket)` returns the user loadcase numbers for the model defined by the variable `Jacket`.

Function: **amcgroupnumber**

Purpose: Returns the user group numbers within a model. Returned as a vector.

Syntax: *amcgroupnumber (model)*

Notes: Group number zero contains all those elements not specified by a group

Example: `Amcgroupelements (Jacket)` returns the user group numbers within the model referred to as `Jacket`.

Function: **amcgroupelements**

Purpose: Returns the user element numbers within a group. Returned as a vector.

Syntax: *amcgroupelements (model, group)*

Notes: Group number zero will return all those elements not specified by a group

Example: `amcgroupelements(Jacket,3)` returns the user element numbers within group 3 of the model referred to as `Jacket`.

Function: **amcloadtimestep**

Purpose: Returns the time steps of the subsets within a load case. Returned as a vector.

Syntax: *amcloadtimestep (model,loadcase)*

Notes: The results will be in the order as they appear in the database

Example: amcloadtimestep(`Jacket`,3) returns the time steps of the subsets of load number 3 of model `Jacket`.

Function: **amcmodeltimestep**

Purpose: Returns the time steps of loads and subsets within a model. Returned as a vector.

Syntax: *amcmodeltimestep (model)*

Notes: The results will be in the order as they appear in the database

Example: amcmodeltimestep(`Jacket`) returns the time steps of the loads and subsets of model denoted by `Jacket`.

Function: **amcnumberincrements**

Purpose: Returns the number of increments within a model. Returned as a scalar.

Syntax: *amcnumberincrements (model)*

Notes: This function is primarily designed to be used with ASAS-NL results.

Example: amcnumberincrements(`Jacket`) returns the number of increments within the model refered to as `Jacket`.

Function: **amcsubsetnumber**

Purpose: Returns the user number of the subsets within a load. Returned as a vector.

Syntax: *amcsubsetnumber (model,load)*

Notes: This function is primarily designed to be used with ASAS-NL results.

Example: amcsubsetnumber(`Jacket`,20) returns the user number of the subsets which constitute load case number 20 of the model refered to as `Jacket`.

4.10 Problem Solving

When using AMC functions, several different types of problem might arise. These fall into three main groups:

- errors in function names and arguments
- errors in values given to a function
- errors in accessing the ASAS files

Some problems may cause Mathcad to display its standard error indicator, which is to highlight the offending function or variable in red. If the erroneous expression is clicked upon an error message will be displayed.

"This variable or function is not defined above" usually means that you have mis-typed the name, or it may be that the amc32.dll and/or the amcinterface.mcd have not been correctly installed.

"This function needs more/less arguments" is self-explanatory. Use the Insert Function dialogue to get information about how many arguments are required for AMC functions.

"This operation can only be performed on an array. It can't be performed on a number" means that the wrong type of arguments have been given (such as text where a number is expected). This is common when passing the model name to a function if the amcinterface.mcd file has not been referenced.

Other problems cause AMC to display one of its own error messages. These messages are generally self-explanatory. A list of these errors is given in the appendix below.

Some general rules for solving problems are:

- Check that the model files are readable by using the amcuserelementlist function. If this cannot retrieve the element list, then either the model data is incorrect or there is something wrong with the ASAS files.
- If an error message is described as "unable to open ... file" this usually means that something in the first part of the function call is wrong: perhaps the directory name is wrong, or the project or structure name. Perhaps the files have been deleted!
- If the ASAS analysis has been re-run since the worksheet was last used, make a null edit of the variable defining the ASAS model and Mathcad will re-calculate as if you had changed the model (even though you know that you have not). This may clear up problems due to Mathcad displaying out-of-date results.

  The individual functions in the DLL are designed to be fairly robust so they will not cause Mathcad or Windows to crash in the event of problems. They only read from ASAS files, so cannot corrupt any data. It is, however, not advisable to attempt to read from an ASAS backing file at the same time as an ASAS program is trying to write to it.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.11 Restrictions

All results should be checked by the user.

There are no facilities for writing information back to the ASAS files.

The DLL functions can be accessed directly from Mathcad, bypassing the amcinterface.mcd. However, this is not recommended because of the need to pass strings as numerical vectors, which requires the use of the Mathcad functions vec2str and str2vec.

4.12 Performance

The AMC functions are designed for convenience, not speed. The functions in the DLL do include some local caching to reduce the amount of file access when a single model is being used. However they do perform a lot of input from the ASAS file system to retrieve even small amounts of data, since they comply with the general Windows recommendation for not keeping files open between calls (which also means that they do not require initialisation or close down and should not cause system problems if the ASAS files are updated while Mathcad is running),.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.13 Model Updates

Like all spreadsheet programs, Mathcad tries to avoid doing unnecessary recalculation. If nothing has changed in a spreadsheet, it does not automatically recalculate it.

This can cause a problem with ASAS results where models may have been rerun. As far as Mathcad is concerned, a function asking for (say) the forces in member 123 at node 567 for load case 89 does not need recalculating if none of these numbers have changed. It does not take account of the fact that the data in files outside Mathcad may have changed.

To force all the results to be reloaded, select Math | Calculate Worksheet from the main menu bar. Mathcad will recalculate the worksheet and update all the values.

Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.